STUDY ON SUPPORT VECTOR MACHINES AS A CLASSIFIER

A THESIS SUBMITTED IN PARTIAL REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING
BY

RAJKISHORE HEMBRAM
ROLL NO. 107EC002

UNDER THE GUIDANCE OF

Dr. SAMIT ARI
ASSISTANT PROFESSOR

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA
ORISSA 769008
INDIA

National Institute of Technology
Rourkela

CERTIFICATE

*This is to certify that the thesis entitled, "**STUDY ON SUPPORT VECTOR MACHINE AS A CLASSIFIER**" submitted by **RajkishoreHembram** in partial fulfillment of requirements for the award of BachelorsofTechnology degree in Electronics and Communication Engineering, Department of Electronics and Communication Engineering at National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.*

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any University/Institute for award of any degree or diploma.

Date: 13th May, 2011
Place: Rourkela

Dr. SamitAri
Assistant Professor
Dept. of Electronics & Communication Engg.
National Institute of Technology.
Rourkela
Orissa- 769008

ACKNOWLEDGEMENT

Ihave this opportunity to thank all the concern individuals whose guidance, help and timely support made me to complete this project within the stipulated time. Unfortunately, it isnot possible to express my thanks to all of them in a single page of acknowledgement.

First and foremost I would like to express my sincere gratitude to my project supervisor Dr. Samit Ari for his great support, nice motivation, proper guidance and constant encouragement which made this project enrich and insightful experience. His timely inputs, valuable return and constructive criticism in many stages of this work was catalyst in understanding and execution of this project.

I am also grateful to Prof. S.K. Patra (Head of the Department), Department of Electronics and Communication Engineering for assigning me the real life application based project and providing me with various facilities of the department. His valuable suggestions and inspiring guidance was extremely helpful.

An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. I acknowledge our indebtedness to all of them.

I would also like to thank all professors and lecturers, and members of the department for their generous help in various ways for the completion of this thesis. I also extend our thanks to our dear friends for their support and cooperation.

Date: 13th May 2011
Place: NIT Rourkela

RajkishoreHembram
Dept. of ECE Engineering
National Institute of Technology
Rourkela – 769008

CONTENTS

Abstract

SVM [1], [2] is a learning method which learns by considering data points to be in space. We studied different types of Support Vector Machine (SVM). We also observed their classification process. We conducted10-fold testing experiments on LSSVM [7], [8] (Least square Support Vector Machine) and PSVM [9] (Proximal Support Vector Machine) using standard sets of data. Finally we proposed a new algorithm NPSVM (Non-Parallel Support Vector Machine) which is reformulated from NPPC [12], [13] (Non-Parallel Plane Classifier). We have observed that the cost function of NPPC is affected by the additional constraint for Euclidean distance classification. So we implicitly normalized the weight vectors instead of the additional constraint. As a result we could generate a very good cost function. The computational complexity of NPSVM for both linear and non-linear kernel is evaluated. The results of 10-fold test using standard data sets of NPSVM are compared with the LSSVM and PSVM.

Notations and terms

These are some words about our notation used in our paper. All vectors will be treated as column vectors unless transposed to a row vector denoted by a prime superscript T. The inner (scalar) product of two column vectors x and y in the real n-dimensional space R^n will be denoted by $x^T y$, and ||x|| will denote the 2-norm of x. For the matrix $A \in R^{mxn}$; A_i will be the i^{th}row of A which will be a row vector in R^n. A column vector of ones of arbitrary suitable dimension will be denoted by column matrix e and the identity matrix of arbitrary suitable order will be denoted by I. The gradient of a differentiable function f on R^n is denoted as:

$\nabla f = [\frac{df}{dx_1}, \frac{df}{dx_2}, \dots, \frac{df}{dx_n}]$ and the hessian if the function f is denoted by $\nabla^2 f = \begin{bmatrix} \frac{d^2 f}{dx_1{}^2} & \cdots & \frac{d^2 f}{dx_n dx_1} \\ \vdots & \ddots & \vdots \\ \frac{d^2 f}{dx_1 dx_n} & \cdots & \frac{d^2 f}{dx_n{}^2} \end{bmatrix}$. The kernel function is denoted by $\varphi(x)$. The Gaussian kernel function is denoted by K(x,C) where C is the set of centers.

Index terms

Support vector machine (SVM) [1], [2] – It is learning method which learns by considering data points to be in space.

Hyper plane– It is the flat surface of (n-1) dimension in an n-dimensional space.

p-value [21] – It is probability of observed or larger difference between two test sets of correctness value, with an assumption of the null hypothesis that there is zero difference between the test sets.

10-fold test – It is a testing process which is repeated 10 times and changing the 10% of data used for testing each time taking the remaining 90% data as training data.

List of figures **Page no**

List of tables

CHAPTER 1
Introduction

1.1 Support vector machine

A support vector machine (SVM) [1], [2] is a supervised learning system based on statistical learning theory. After its introduction, SVM has outperformed most other systems in a wide range of practical applications within a few years. They proved to be excellent tools in the way of analyzing data and recognizing patterns for as well as regression analysis. Vladimir Vapnik invented the original SVM [1] algorithm. The SVM takes in a set of input data and then predicts their corresponding classes for each given input. In this model, SVM represents the data points as points in space, mapped such that the data points of the different categories are separated by a clear and wider gap. This model constructs a hyper plane or set of hyper planes which can be used for classification and regression analysis.

For a linearly separable two class problem, SVM finds an optimal hyper plane that maximizes the separation between the two classes and hence lower the generalization error. For nonlinearly separable case the input data are projected into another high dimensional feature space which makes the data separable in that space. After that SVM is again used to classify the data in the new feature space.

The performance and efficiency of the SVM classifier depends upon the optimal tuning parameters which are usually chosen by cross-validation method. The training time also makes it difficult to handle large sets of optimal parameters. To reduce these computational complexities many types of SVMs are being developed.

1.2 Modified SVMs

Glenn Fung and Olvi L. Mangasarian proposed Proximal Support Vector Machine (PSVM) [9] that contained the idea that we can construct SVMs by assigning one dataset closest to one of the hyper planes. After that Olvi L. Mangasarian and Edward W. Wild came up with an idea that instead of dividing the space into regions for each class, classification will be based on the proximity to the hyper planes. They formulated the Generalized Eigen-value Proximal Support Vector Machine (GEPSVM) [11]. For binary classification, they created two hyper planes instead of one and then assigned classes to different datasets according to their relative proximity to the two hyper planes. It was based on Fisher Information matrix [23], [24]. Jayadeva et al. proposed Twin Support Vector Machine [10] (TWSVM) where same proximity rule of GEPSVM was used. It solves two Quadratic Programming Problems (QPPs) unlike the GEPSVM. Their results seemed to be more accurate but the complexity was more. Further their classification was not completely according to the distance of the datasets to the hyper planes. Santanu Ghorai et al. proposed Nonparallel Plane Proximal Classifier [12], [13] (NPPC) where they further improvised the TWSVM and made the computational complexity less. They replaced the inequality constraints with the equality constraints. Their method is based on the Euclidean distance which makes it a proximal classifier in real sense. The only disadvantage with their classification process is their tuning process which again increases the overall time.

1.3 Objective:

Our main objective is to study different types of support vector machines and determine the performance of the earlier SVMs. We propose a new type of binary class data classifier, named as Non-Parallel Support Vector Machine (NPSVM).

CHAPTER 2
Database and Experimental setup

2.1 Data

Some standard sets of data are collected from UCI Repository [19] for the purpose of our experiments.

Table 2.1 Information about the data sets used for numerical testing and comparison

Data sets	Class Ratio	Description
Australian (690x14)	307:383	Data sets containing information for approval of Australian credit card
Bupa Liver (345x16)	200:145	Data sets containing sensitive blood test results for liver disorder
German (1000x24)	700:300	Data sets containing information for approval of German credit card
Heart-Stat log (270x13)	150:120	Data sets containing information for detecting heart disease
Ionosphere (351x34)	225:126	Data sets containing information for detecting good or bad RADAR signal
Pima Indian (768x8)	500:268	Data sets containing information for detecting diabetes for patients of Pima Indian heritage
WDBC (569x31)	357:212	Data sets containing information about the fine needle aspirate of breast cancer from digitized image
WPBC (148x32)	151:27	Data sets containing follow up information about the fine needle aspirate of breast cancer from digitized image

2.2 Experimental Setup

We have taken different sets of publicly available benchmark data sets from UCI Repository [19] for the purpose of testing. All the numerical testing has been done using MATLAB 7.6 Version on Windows 7 operating system on a CPU with an i5 processor with speed of 3.33 GHz and 4 GB RAM. We have tested NPSVM, PSVM and LS-SVM for both the linear and the Gaussian kernel. The codes for PSVM and LS-SVM are taken from the SVM Tool Box [16] and LSSVM Tool Box [17]. The implementation of NPSVM is done using simple 'cgs' MATLAB function.

CHAPTER 3
Earlier SVMs and their Characteristics

3.1 Least Square Support Vector Machine (LSSVM)

The major drawback of the original SVM is that it takes a large amount of training time. Suykens et al. resolved this problem by introducing LS-SVM [7], [8]. They reformulated the classification problem as:

$$\min_{w,b,\xi} J(w,b,\xi) = \{\frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{m}{\xi_i}^2\}$$

`

such that $y_i(w^T\varphi(x_i) + b) = 1 - \xi_i$ (3.1)

In this case, the inequality constraint condition used for original SVM is replaced by the equality constraints. This reduces the problem to m-number of linear equations. The computational complexity reduces drastically as well as gives approximately the same accuracy. The Lagrangian of the cost function can be defined as:

$$L(w,b,\xi:\alpha) = J(w,b,\xi) - \sum_{i=1}^{m}\alpha_i[y_i(w^T\varphi(x_i) + b) + 1 - \xi_i]$$ (3.2)

α_i are the Lagrangian multipliers due to equality constraints which is proportional to errors. For the conditions of optimality the Lagrangian is differentiated with respect to w, b, ξ and α which gives:

$$\frac{dL}{dw} = 0 \rightarrow w = \sum_{i=1}^{m}\alpha_i\, y_i\, \varphi(x_i)$$

$\frac{dL}{db} = 0 \rightarrow \sum_{i=1}^{m} \alpha_i \, y_i = 0$

$\frac{dL}{d\xi_i} = 0 \rightarrow \xi_i = \gamma y_i$

$\frac{dL}{d\alpha_i} = 0 \rightarrow [y_i(w^T \varphi(x_i) + b) + 1 - \xi_i] = 0, where \; i = 1,2,..m$

The final solution used for making decision is

$$f(x) = sgn(w^T \varphi(x) + b) \tag{3.3}$$

Experiments and Results

The LSSVM is implemented by using the LSSVM Tool Box [17]. We have taken γ value as 10 for the 10-fold testing purpose. The accuracy (in percentage), standard deviation and the time taken for 10-fold test is calculated.

Table 3.2.1 Results for LSSVM

DATASETS		LSSVM
Australian	Accuracy	86.08696
(690x14)	Standard deviation	5.768043
	time	1.294808
Bupa Liver	Accuracy	64.61345
(345x6)	Standard deviation	11.99666
	time	0.374402
German	Accuracy	76.2
(1000x24)	Standard deviation	5.878775
	time	2.402415
Heart-Stat log	Accuracy	83.33333
(270x13)	Standard deviation	3.414646
	time	0.390003
Ionosphere	Accuracy	85.42857
(351x34)	Standard deviation	10.57143
	time	0.936006
Pima Indian	Accuracy	77.20267
(768x8)	Standard deviation	5.420182
	time	1.216808
WDBC	Accuracy	95.4198
(569x31)	Standard deviation	3.888999
	time	1.201208
WPBC	Accuracy	77.25146
(148x32)	Standard deviation	10.97937
	time	0.686404

3.2 Proximal Support Vector Machine (PSVM)

In PSVM [9] instead of dividing the space into disjoint regions for each class, the data points are assigned according to the proximity to the hyper planes that are pushed apart as far as possible [9]. This leads to a very fast and simple algorithm. The cost function is given as follows:

$$\min_{w,b,\xi} J(w,b,\xi) = \{\frac{1}{2}||[w,b]^T||^2 + C\sum_{i=1}^{m}{\xi_i}^2\}$$

Such that $y_i(w^T\varphi(x_i)+b) = 1-\xi_i$ (3.4)

The minimization of cost function leads to maximization of margin in $[w\ b]$ space.It also uses the equality constraint and minimizes the squared error like LS-SVM. The PSVM works much faster than SVM as well as give performance similar to SVM. The lagrangian of the cost function is given as:

$$L(w,b,\xi:\alpha) = J(w,b,\xi) - \sum_{i=1}^{m}\alpha_i[y_i(w^T\varphi(x_i)+b)+1-\xi_i] \quad (3.5)$$

α_i are the lagrangian multipliers due to equality constraints which is proportional to errors. The cost function is differentiated with respected to w, b, ξ and α which gives:

$$\frac{dL}{dw} = 0 \rightarrow w = \sum_{i=1}^{m}\alpha_i\, y_i\, \varphi(x_i) \quad ; \quad \frac{dL}{db} = 0 \rightarrow b = \sum_{i=1}^{m}\alpha_i\, y_i \quad ; \quad \frac{dL}{d\xi_i} = 0 \rightarrow \sum_{i=1}^{m}\alpha_i\, y_i = 0$$

$$\frac{dL}{d\alpha_i} = 0 \rightarrow [y_i(w^T\varphi(x_i)+b)+1-\xi_i] = 0, where\ i = 1,2,..m$$

The decision for classification is based on equation (3.2).

Experiment and Results

The PSVM was implemented using the PSVM Tool Box [16]. Unlike LSSVM, PSVM was subjected to tuning before training. The total time taken by PSVM is the time taken for 10-fold testing and for tuning to the regularization parameter. The result is compared with the LSSVM.

Table 3.3.1 Results of PSVM and comparison with LSSVM

DATASETS		PSVM	LSSVM
Australian	Accuracy	85.94203	**86.08696**
(690x14)	Standard deviation	5.689213	**5.768043**
	time	0.124801	**1.294808**
Bupa Liver	Accuracy	64.03361	**64.61345**
(345x6)	Standard deviation	13.70882	**11.99666**
	time	0.0624	**0.374402**
German	Accuracy	75.8	**76.2**
(1000x24)	Standard deviation	5.946427	**5.878775**
	time	0.109201	**2.402415**
Heart-Stat log	Accuracy	**84.44444**	83.33333
(270x13)	Standard deviation	**2.771598**	3.414646
	time	**0.0624**	0.390003
Ionosphere	Accuracy	85.14286	**85.42857**
(351x34)	Standard deviation	10.82703	**10.57143**
	time	0.0624	**0.936006**
Pima Indian	Accuracy	76.94293	**77.20267**
(768x8)	Standard deviation	5.533878	**5.420182**
	time	0.078	**1.216808**
WDBC	Accuracy	94.71178	**95.4198**
(569x31)	Standard deviation	4.689397	**3.888999**
	time	0.0624	**1.201208**
WPBC	Accuracy	75.05848	**77.25146**
(148x32)	Standard deviation	10.34282	**10.97937**
	time	0.078	**0.686404**

The best accuracy are denoted in bold.

Conclusion: It is seen that the PSVM solves the problem faster than LSSVM but the performance of the classifier (i.e. accuracy) decreases in most of the case.

CHAPTER 4
The linear Non-Parallel Support Vector Machine (NPSVM)

4.1 Formulation of linear NPSVM

We will discuss about the formulation of the NPSVM in this section. The concepts used for NPPC [12], [13] are still in use with some modifications in the cost function. The datasets belonging to one of the class are assigned to matrix A and datasets belonging to the other class are assigned to matrix B. The new cost function according to our new model is:

$$\min_{\hat{w}_1, b_1} V_1(\hat{w}_1, b_1) = \frac{1}{2}\left|\left|A\hat{w}_1 + e_1 b_1\right|\right|^2 + e_2{}^T \xi_2 + \frac{1}{2}||\xi_2||^2$$

Such that $-(B\hat{w}_1 + e_2 b_1) + \xi_2 = e_2, \xi_2 \geq 0$ (4.1)

And

$$\min_{\hat{w}_2, b_2} V_2(\hat{w}_2, b_2) = \frac{1}{2}\left|\left|B\hat{w}_2 + e_2 b_2\right|\right|^2 + e_1{}^T \xi_1 + \frac{1}{2}||\xi_1||^2$$

Such that $(A\hat{w}_2 + e_1 b_2) + \xi_1 = e_1, \xi_1 \geq 0$ (4.2)

The advantage of this formulation over NPPC is that we have dropped the idea of using different regularization parameters. All the terms in the cost function are given equal importance. The first term is used for reducing the Euclidean distance of the one of the hyper plane to remain closer to the corresponding class of data. The second term and third term are to reduce the generalized error. The reduction of the error here means to keep the other class datasets in one side of the hyper plane.

The penalty parameter α as in case of NPPC is not used here because we are directly using unity norm weight matrices instead of normal weights. This makes our cost function free from any constraints affecting the cost function.

Unlike TWSVM, NPSVM classifies different datasets according to the Euclidean distance from the hyper planes. At the same time NPSVM is as faster as the LSSVM, GEPSVM, NPPC and PSVM. Since the tuning (regularization) sets of NPPC is very large and again the optimal parameters are to be found by cross validation, the overall time to train it becomes very high.

We will be using only the conjugate gradient method for obtaining the optimal hyper planes. First of all we need to find out the gradient matrix g and the hessian matrix H for both the cost functions.

The cost function V_1 can be written as:

$$V_1(\hat{w}_1, b_1) = \frac{1}{2}\left|\left|A\hat{w}_1 + e_1 b_1\right|\right|^2 + e_2{}^T[e_2 + (B\hat{w}_1 + e_2 b_1)] + \frac{1}{2}||e_2 + (B\hat{w}_1 + e_2 b_1)||^2 \quad (4.3)$$

Now for construction of the hyper plane 1

$$\hat{w}_1{}^T x + b_1 = 0 \quad (4.4)$$

let us define

$L = [A\ e_1]$,

$M = [B\ e_2]$,

Now the first order partial derivative with respect to $[\hat{w}_1, b_1]$ of the cost function V_1 is given as: $g_1 = \nabla V_1 = [L^T L + M^T M] \begin{bmatrix} \hat{w}_1 \\ b_1 \end{bmatrix} + 2M^T e_2$ (4.5)

And the second order partial derivative of V_1 is given as:

$$H_1 = \nabla^2 V_1 = [L^T L + M^T M] \quad (4.6)$$

is a positive constant;

The positive second derivative ensures minima for our cost function.

Similarly for determination of second hyper plane of the second cost function V_2

The cost function V_2 can be written as:

$$V_2(\hat{w}_2, b_2) = \frac{1}{2}\left||B\hat{w}_2 + e_2 b_2|\right|^2 + {e_1}^T[e_1 - (A\hat{w}_2 + e_1 b_2)] + \frac{1}{2}\left||e_1 - (A\hat{w}_2 + e_1 b_2)|\right|^2 \quad (4.7)$$

Now for constructing the hyper plane 2

$${\hat{w}_2}^T x + b_2 = 0 \quad (4.8)$$

The first order partial derivative with respect to $[\hat{w}_2, b_2]$ of the cost function V_2 is evaluated:

$$g_2 = \nabla V_2 = [M^T M + L^T L] \begin{bmatrix} \hat{w}_2 \\ b_2 \end{bmatrix} - 2L^T e_2 \quad (4.9)$$

And the second order partial derivative of V_1 is derived:

$$H_2 = \nabla^2 V_2 = [M^T M + L^T L] \quad (4.10)$$

is a positive constant.

In order to get the hyper planes 1 and 2 shown in equation (4.4) and (4.8) respectively, we solve equation (4.5) and (4.9) by equating them to zero to get the minima of the respective cost functions. The method used to solve the two equations is the conjugate gradient method.

4.2 Conjugate gradient method

It is the method in which the nearest local minimum of a function with n variables is evaluated. Instead of using local gradient for going downhill, it uses conjugate directions [20] to approach the local minimum.

4.3 Classification

The data sets are assigned classes exactly according to their Euclidean distance from the hyper plane in the [w] space. The classifying function used for NPSVM is given as:

$$Class\ k = \min_{k=1,2} \frac{|{w_k}^T x + b_k|}{|||[w_k]|||} \tag{4.11}$$

4.4 Computational complexity and efficient algorithm for NPSVM

Let m_1= number of datasets belonging to class 1,

m_2 = number of datasets belonging to class 2.

And $m = m_1 + m_2$

Let N be the maximum no of iterations to be used for attaining convergence

Algorithm for finding out hyper plane 1

1) Set convergence parameter ε and maximum no of iterations N
2) Compute $H_1 = P^T P + Q^T Q$ and $g_{10} = 2Q^T e_2$

3) Initialize $u = [w_1\ b_1]^T$
4) Compute $||w_1||$ and take $u_1 = u/||w_1||$
5) Solve $H_1 u_1 = -g_{10}$ by using conjugate gradient method (with ε as the convergence parameter and N as the maximum iteration).

Costs involved for hyper plane 1

Step1 and step 3 has negligible costs.

Step 2 cost is $O\ (m_1 n^2 + m_2 n^2 + m_2 n) = O(mn^2 + m_2 n)$,

Step 4 cost is $O(n + n) = O(2n)$

Step 5 cost is $O(ln^2)$ where l the no of iterations is taken by the conjugate gradient method to converge to ε, for extreme case cost is $O(Nn^2)$

The total computation cost for all the operations become $O(mn^2 + m_2 n + 2n + Nn^2)$

Algorithm for finding out hyper plane 2

6) Set convergence parameter ε and maximum no of iterations N
7) Compute $H_2 = Q^T Q + P^T P$ and $g_{20} = 2P^T e_1$
8) Initialize $u = [w_2\ b_2]^T$
9) Compute $||w_2||$ and take $u_2 = u/||w_2||$
10) Solve $H_2 u_2 = -g_{20}$ by using conjugate gradient method (with ε as the convergence parameter and N as the maximum iteration).

Costs involved for hyper plane 2

Step 6 and step 8 has negligible costs.

Step 7 cost is $O\ (m_2n^2 + m_1n^2 + m_1n) = O(mn^2 + m_1n)$,

Step 9 cost is $O(n + n) = O(2n)$

Step 10 cost is $O(ln^2)$ where l is the no. of iterations taken by the conjugate gradient method to converge to ε. For extreme case, cost becomes $O(Nn^2)$

The total computational cost becomes $O(mn^2 + m_1n + 2n + Nn^2)$

And hence the net complexity becomes

$$O(2mn^2 + mn + 4n + 2Nn^2)$$

Since $H_1 = H_2$, there is no need not for extra calculation of H_2 and hence the optimum complexity becomes

$$O(mn^2 + mn + 4n + 2Nn^2)$$

4.5 Experiment and Results

We have conducted two sets of experiments. In the first set we have used the data in the same order as given in the dataset for the purpose of testing. In the second case we shuffled the data in order to get the maximum information content for the training process. The results of NPSVM are compared with the PSVM and LSSVM. The p-value of the paired t-test [21] is calculated for every 10-fold testing.

Table 4.5.1 Results of 10-fold accuracy according linear kernel with training without shuffling

DATASETS		NPSVM (Proposed)	PSVM	LSSVM
Australian	Accuracy	85.94203	85.94203	**86.08696**
(690x14)	Standard deviation	5.689213	5.689213	**5.768043**
	time	0.327602	0.124801	**1.294808**
	p-value		1	**0.316643**
Bupa Liver	Accuracy	**64.61345**	64.03361	**64.61345**
(345x6)	Standard deviation	**11.99666**	13.70882	**11.99666**
	time	**0.202801**	0.0624	**0.374402**
	p-value		0.624246	**1**
German	Accuracy	76.1	75.8	**76.2**
(1000x24)	Standard deviation	6.057227	5.946427	**5.878775**
	time	0.858005	0.109201	**2.402415**
	p-value		0.169257	**0.316643**
Heart-Stat log	Accuracy	83.33333	**84.44444**	83.33333
(270x13)	Standard deviation	3.414646	**2.771598**	3.414646
	time	0.202801	**0.0624**	0.390003
	p-value		**0.065262**	1
Ionosphere	Accuracy	**85.42857**	85.14286	**85.42857**
(351x34)	Standard deviation	**10.57143**	10.82703	**10.57143**
	time	**0.296402**	0.0624	**0.936006**
	p-value		0.570076	**1**
Pima Indian	Accuracy	**77.20267**	76.94293	**77.20267**
(768x8)	Standard deviation	**5.420182**	5.533878	**5.420182**
	time	**0.327602**	0.078	**1.216808**
	p-value		0.144928	**1**
WDBC	Accuracy	**95.77381**	94.71178	95.4198
(569x31)	Standard deviation	**3.632531**	4.689397	3.888999
	time	**0.374402**	0.0624	1.201208
	p-value		0.268201	0.144945
WPBC	Accuracy	77.19298	75.05848	**77.25146**
(148x32)	Standard deviation	13.16959	10.34282	**10.97937**
	time	0.156001	0.078	**0.686404**
	p-value		0.466482	**0.975276**

The figures which written are in bold shows the best accuracy.

Table 4.5.2 Results of 10-fold accuracy according linear kernel with training along with shuffling

DATASETS		NPSVM (Proposed)	PSVM	LSSVM
Australian	Accuracy	**93.47826**	**93.47826**	**93.47826**
(690x14)	Standard deviation	**1.485065**	**1.485065**	**1.485065**
	time	**0.296402**	**0.109201**	**1.216808**
	p-value		**1**	**1**
Bupa Liver	Accuracy	**98.84034**	95.63025	**98.84034**
(345x6)	Standard deviation	**1.910652**	6.972157	**1.910652**
	time	**0.156001**	0.093601	**0.343202**
	p-value		0.147045	**0.984052**
German	Accuracy	**95.7**	95.3	**95.7**
(1000x24)	Standard deviation	**1.486607**	1.268858	**1.486607**
	time	**0.826805**	0.0624	**2.433616**
	p-value		0.316643	**1**
Heart-Stat log	Accuracy	97.77778	**98.88889**	97.77778
(270x13)	Standard deviation	2.962963	**2.371527**	2.962963
	time	0.187201	**0.0624**	0.421203
	p-value		**0.169257**	1
Ionosphere	Accuracy	**100**	**100**	**100**
(351x34)	Standard deviation	**0**	**0**	**0**
	time	**0.249602**	**0.0624**	**0.889206**
	p-value		**1**	**0.316643**
Pima Indian	Accuracy	**73.83459**	**73.83459**	**73.83459**
(768x8)	Standard deviation	**3.316452**	**3.316452**	**3.316452**
	time	**0.296402**	**0.0624**	**1.294808**
	p-value		**0.316643**	**1**
WDBC	Accuracy	**98.94737**	96.49123	**98.94737**
(569x31)	Standard deviation	**2.625724**	3.508772	**2.625724**
	time	**0.468003**	0.0624	**1.232408**
	p-value		7.74E-06	**1**
WPBC	Accuracy	**98.42105**	93.09942	96.84211
(148x32)	Standard deviation	**4.736842**	7.813312	9.473684
	time	**0.187201**	0.078	0.686404
	p-value		0.016811	0.316643

The figures which written are in bold shows the best accuracy.

The results show that NPSVM takes less time in comparison of LSSVM for solving linear problems with almost same performance (i.e. accuracy).

CHAPTER 5
Non-linear kernel NPSVM

5.1 Kernel Function

When the data becomes inseparable we use other kernel [9], [14], [15] functions instead of the linear kernel one for better classification. We are using Gaussian radial basis function [2] as a kernel for the non-linear case. Instead of classifying data according their attributes, some datasets from any one class are chosen as centers for classification. The classification is based on the distance of the data points from the centers.

The kernel function used in our case is given by:

$$K(x, C') = e^{-\mu \Sigma (x - x_i)^2} \quad (5.1)$$

where the parameter $\mu(mu) = 1/2\sigma^2$

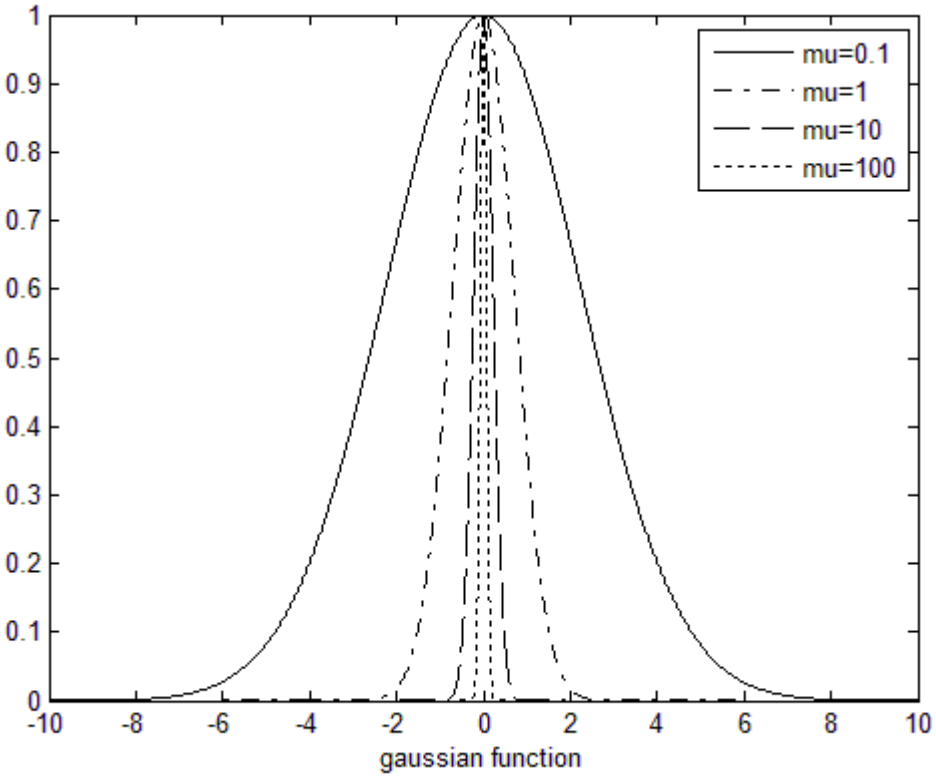

Figure 5.1.1

Therefore larger the value of μ is chosen, smaller will be the variance and hence stricter will be the kernel, and when smaller value of μ is taken smaller, larger will be the variance and hence the kernel will become more liberal.

The data becomes highly separable in the higher dimension. After conversion same formulations are used which are already used in case of linear kernel.

5.2 Formulation of non-linear NPSVM

The cost function V_{K_1} can be written as:

$$V_{K_1}(\hat{w}_1, b_1) = \frac{1}{2}\left|\left|K(A, C')\hat{w}_1 + e_1 b_1\right|\right|^2 + e_2{}^T[e_2 + (K(B, C')\hat{w}_1 + e_2 b_1)] + \frac{1}{2}\left|\left|e_2 + (K(B, C')\hat{w}_1 + e_2 b_1)\right|\right|^2 \quad (5.2)$$

Now for construction of the hyper plane 1

$$\hat{w}_1{}^T K(x, C') + b_1 = 0 \quad (5.3)$$

let us define

$L_K = [K(A, C')e_1]$,

$M_K = [K(B, C')e_2]$,

Now the first order partial derivative with respect to $[\hat{w}_1, b_1]$ of the cost function V_1 is given as:

$$g_{K_1} = \nabla V_{K_1} = \left[L_K{}^T L_K + M_K{}^T M_K\right]\begin{bmatrix}\hat{w}_1\\ b_1\end{bmatrix} + 2M_K{}^T e_2 \tag{5.4}$$

And the second order partial derivative of V_1 is given as:

$$H_{K_1} = \nabla^2 V_{K_1} = \left[L_K{}^T L_K + M_K{}^T M_K\right] \tag{5.5}$$

is a positive constant;

The positive second derivative ensures minima for our cost function.

Similarly for determination of second hyper plane of the second cost function V_2

The cost function V_{K_2} can be written as:

$$V_{K_2}(\hat{w}_2, b_2) = \frac{1}{2}\left|\left|K(B, C')\hat{w}_2 + e_2 b_2\right|\right|^2 + e_1{}^T[e_1 - (K(A, C')\hat{w}_2 + e_1 b_2)] + \frac{1}{2}\left|\left|e_1 + (K(A, C')\hat{w}_2 + e_1 b_2)\right|\right|^2 \tag{5.6}$$

Now for constructing the hyper plane 2

$$\hat{w}_2{}^T K(x, C') + b_2 = 0 \tag{5.7}$$

The first order partial derivative with respect to $[\hat{w}_2, b_2]$ of the cost function V_2 is evaluated:

$$g_{K_2} = \nabla V_{K_2} = \left[M_K{}^T M_K + L_K{}^T L_K\right]\begin{bmatrix}\hat{w}_1\\ b_1\end{bmatrix} - 2L_K{}^T e_2 \tag{5.8}$$

And the second order partial derivative of V_1 is derived:

$$H_{K_2} = \nabla^2 V_{K_2} = \left[M_K{}^T M_K + L_K{}^T L_K\right] \tag{5.9}$$

is a positive constant.

In order to get the hyper planes 1 and 2 shown in equation (5.3) and (5.7) respectively, we have to solve equation (5.4) and (5.8) by equating them to zero to get the minima of the respective cost functions. We again use the conjugate gradient method to solve the two equations.

5.3 Classification:

The classification of the data points is done in the high dimension feature space. The classification is again done according to the Euclidean distance from the hyper planes. The function used to classify the data is given as:

$$Class\ k = \min_{k=1,2} \frac{|w_k{}^T K(x,C^{'})+b_k|}{||[w_k]||} \qquad (5.10)$$

5.4 Computational complexity and efficient algorithm for non-linear kernel NPSVM

For converting the inputs to a higher dimension, we choose datasets of A class as the center.The complexity for conversion will be:

$O(m_1{}^2 n^2)$for matrix A to $K(A,C^{'})$and $O(m_1 m_2 n^2)$ for matrix B to $K(B,C^{'})$.

Here the dimension of the input matrix changes to $m \times m1$

Let N be the maximum no of iterations to be used for attaining convergence.

Algorithm for finding out hyper plane 1

1) Set convergence parameter ε and maximum no of iterations N
2) Compute $H_{K_1} = L_K{}^T L_K + M_K{}^T M_K$ and $g_{K_{10}} = 2M_K{}^T e_2$

3) Initialize $u = [w_1\ b_1]^T$
4) Compute $||w_1||$ and take $u_1 = u/||w_1||$
5) Solve $H_{K_1} u_1 = -g_{K_{10}}$ by using conjugate gradient method (with ε as the convergence parameter and N as the maximum iteration).

Costs involved for hyper plane 1

Step 6 and step 8 has negligible costs.

Step 7 cost is $O\ ({m_1}^3 + m_2 {m_1}^2 + m_2 m_1) = O(m{m_1}^2 + m_2 m_1)$,

Step 9 cost is $O(m_1 + m_1) = O(2m_1)$

Step 10 cost is $O(l{m_1}^2)$ where l the no of iterations is taken by the conjugate gradient method to converge to ε, for extreme case cost is $O(N{m_1}^2)$

The total computation cost for finding solutions to operations become

$$O(m{m_1}^2 + m_2 m_1 + 2m_1 + N{m_1}^2)$$

Algorithm for finding out hyper plane 2

6) Set convergence parameter ε and maximum no of iterations N
7) Compute $H_{K_2} = {M_K}^T M_K + {L_K}^T L_K$ and $g_{K_{20}} = -2{L_K}^T e_2$
8) Initialize $u = [w_2\ b_2]^T$
9) Compute $||w_2||$ and take $u_2 = u/||w_2||$
10) Solve $H_{K_2} u_2 = -g_{K_{20}}$ by using conjugate gradient method (with ε as the convergence parameter and N as the maximum iteration).

Costs involved for hyper plane 2

Step 6 and step 8 has negligible costs.

Step 7 cost is $O\ (m_2{m_1}^2 + {m_1}^3 + {m_1}^2) = O(m{m_1}^2 + {m_1}^2)$,

Step 9 cost is $O(m_1 + m_1) = O(2m_1)$

Step 10 cost is $O(l{m_1}^2)$ where l is the total no of iterations taken by the conjugate gradient method to converge to □. For extreme case, cost becomes $O(N{m_1}^2)$

The total computational cost becomes $O(m{m_1}^2 + {m_1}^2 + 2m_1 + N{m_1}^2)$

And hence the net complexity becomes

$$O(2m{m_1}^2 + m{m_1} + 4m_1 + 2N{m_1}^2)$$

Since $H_{K_1} = H_{K_2}$, there is no need not calculate □$_2$ and hence the optimum complexity becomes

$$C_{m1} = O(m{m_1}^2 + m{m_1} + 4m_1 + 2N{m_1}^2)$$

The overall complexity becomes

$$(C_{m1} + cost\ for\ conversion) = O(m{m_1}^2 + m{m_1} + 4m_1 + 2N{m_1}^2 + {m_1}^2n^2 + m_1m_2n^2)$$

5.5 Reduced kernel

We decrease the no of centers arbitrarily to m_1' instead of m_1 , thus reducing the computational cost.

Let $m_1 = r * m_1$' where m_1'is an integer and $r \geq 1$

For defining the input kernels the complexity becomes

$O(m_1 m_1' n^2)$ for class 1 and $O(m1'm2n2)$for class 2 which is □ times less than complete kernels.

The net computational complexity becomes

$$C_{m2} = O(mm_1'^2 + mm_1' + 4m_1' + 2Nm_1'^2)$$

And the overall complexity becomes

$$(C_{m2} + cost\ for\ conversion)$$
$$= O(mm_1'^2 + mm_1' + 4m_1' + 2Nm_1'^2 + m_1'^2 n^2 + m_1' m_2 n^2)$$

For the extreme case:

Now

$$C_{m1} \approx O(mm_1{}^2 + 2Nm_1{}^2)$$

And

$$C_{m2} \approx O(mm_1'^2 + 2Nm_1'^2)$$

That implies $C_{m2} \approx {C_{m1}}/{r^2}$

Hence Complexity is reduced to r^2 times for a reduced kernel.

5.6 Experiment and Results

We have taken three tests for non-linear kernel. Before doing we shuffled the data for increasing the information in the training process. In the first test, μ value for the Gaussian function is taken 100 and for the second test it is taken 10. In the third case, we took the reduced kernel [22] where we took arbitrary 60% class 1 data as training centers. In this case, we only checked the total time taken by the classifier for each data set.

Table 5.6.1 Results of 10-fold accuracy according non-linear kernel (μ=10) along with shuffling

DATASETS		NPSVM (Proposed)	PSVM	LSSVM
Australian	Accuracy	**99.56522**	99.13043	**99.56522**
(690x14)	Standard deviation	**1.304348**	1.833204	**1.304348**
	time	**3.915625**	6.598842	**8.564455**
	p-value		0.316643	**1**
Bupa Liver	Accuracy	99.42017	93.31933	**99.71429**
(345x6)	Standard deviation	1.159816	8.256727	**0.857143**
	time	0.405603	1.49761	**0.670804**
	p-value		0.02467	**0.316643**
German	Accuracy	**99.7**	**99.7**	99.6
(1000x24)	Standard deviation	**0.640312**	**0.674949**	0.8
	time	**5.101233**	**20.54533**	9.31326
	p-value		**1**	0.316643
Heart-Stat log	Accuracy	**99.62963**	**99.62963**	**99.62963**
(270x13)	Standard deviation	**1.111111**	**1.171214**	**1.111111**
	time	**0.624004**	**1.419609**	**1.762811**
	p-value		**1**	**1**
Ionosphere	Accuracy	98.3254	**98.87302**	97.76984
(351x34)	Standard deviation	4.159586	**2.376186**	5.800979
	time	2.418016	**7.035645**	4.804831
	p-value		**0.540317**	0.316643
Pima Indian	Accuracy	94.91969	**99.60868**	**99.60868**
(768x8)	Standard deviation	2.068926	**0.87854**	**0.833456**
	time	3.260421	**6.24004**	**8.299253**
	p-value		**2.75E-06**	**2.75E-06**
WDBC	Accuracy	98.07018	**98.42105**	**98.42105**
(569x31)	Standard deviation	5.230895	**4.99307**	**4.736842**
	time	1.950012	**9.937264**	**2.979619**
	p-value		**0.144928**	**0.144928**
WPBC	Accuracy	**100**	**100**	**100**
(148x32)	Standard deviation	**0**	**0**	**0**
	time	**0.483603**	**1.57561**	**1.014007**
	p-value		**1**	**1**

The figures which written are in bold shows the best accuracy.

Table 5.6.2 Results of 10-fold accuracy according non-linear kernel (μ=100) along with shuffling

DATASETS		NPSVM (Proposed)	PSVM	LSSVM
Australian	Accuracy	**99.56522**	99.13043	**99.56522**
(690x14)	Standard deviation	**1.304348**	1.833204	**1.304348**
	time	**11.06047**	6.910844	**8.736056**
	p-value		0.316643	**1**
Bupa Liver	Accuracy	**98.55462**	97.10084	**98.55462**
(345x6)	Standard deviation	**2.952631**	5.146025	**2.952631**
	time	**0.374402**	1.435209	**0.670804**
	p-value		0.155118	**1**
German	Accuracy	**99.9**	**99.9**	**99.9**
(1000x24)	Standard deviation	**0.3**	**0.316228**	**0.3**
	time	**12.77648**	**17.51891**	**9.609662**
	p-value		**1**	**1**
Heart-Stat log	Accuracy	**98.88889**	**98.88889**	**98.88889**
(270x13)	Standard deviation	**2.371527**	**2.499809**	**2.371527**
	time	**0.936006**	**1.341609**	**1.778411**
	p-value		**1**	**1**
Ionosphere	Accuracy	97.76984	**98.05556**	97.76984
(351x34)	Standard deviation	5.800979	**6.148873**	5.800979
	time	2.948419	**5.241634**	5.038832
	p-value		**0.316643**	1
Pima Indian	Accuracy	**99.60868**	99.34552	**99.60868**
(768x8)	Standard deviation	**0.833456**	1.415973	**0.833456**
	time	**3.291621**	6.099639	**8.611255**
	p-value		0.316643	**1**
WDBC	Accuracy	**98.42105**	**98.42105**	**98.42105**
(569x31)	Standard deviation	**4.736842**	**4.99307**	**4.736842**
	time	**2.137214**	**10.04646**	**2.901619**
	p-value		**1**	**1**
WPBC	Accuracy	**100**	**100**	**100**
(148x32)	Standard deviation	**0**	**0**	**0**
	time	**0.468003**	**1.466409**	**0.967206**
	p-value		**1**	**1**

The figures which written are in bold shows the best accuracy.

Table 5.5.3 Time taken by the classifiers to train using random 60% of class 1 training centers

Data sets	NPSVM (Proposed)	PSVM	LS-SVM
Australian (690x14)	2.839218	4.68003	5.818837
Bupa Liver (345x16)	0.249602	0.795605	0.468003
German (1000x24)	3.822025	11.76248	6.973245
Heart-Stat log (270x13)	0.546004	0.811205	1.138807
Ionosphere (351x34)	1.684811	3.962425	3.08882
Pima Indian (768x8)	2.402415	4.024826	5.709637
WDBC (569x31)	1.419609	5.491235	2.059213
WPBC (148x32)	0.374402	0.858005	0.733205

When we used kernel technique, the time taken by NPSVM is less than PSVM and LSSVM. In most of the cases NPSVM gave nearly same performance (i.e. accuracy) as PSVM and LSSVM.

CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

- Results show that PSVM is faster than LSSVM but less accurate than LSSVM.
- We have formulated NPSVM which implicitly considers the unity norm equality constraints. The non-linear problems are efficiently solved by projecting the data to a higher dimension. The classifier does not include any regularization coefficients, which makes it spontaneous. It includes the Conjugate Gradient method which increases the computational power of the classifier.
- The accuracy of the classifier is comparable with the LSSVM, but then also the total time taken by the classifier is less than the time taken by LSSVM and PSVM.

6.2 Future Scope

- The NPSVM can be used for classification of multiclass data.
- Instead of using general Conjugate Gradient method for finding solution, we can use other methods which converge much faster and much closer to the solution.

CHAPTER 7
References

[1] C. Cortes, V.N. Vapnik, *Support vector networks, Machine Learning* 20 (3) (1995) 273–297.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*, New York, Springer, 1995.

[3] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines*, vol. 3, Cambridge University Press, Cambridge, MA, 2000 Chapter 6, pp. 113–145.

[4] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining Knowledge Discovery* 2 (2) (1998) 121–167.

[5] N.Cristianini,J. ShaweTaylor, *Kernel Methods for Pattern Analysis*, Cambridge UniversityPress,Cambridge,UK,2004.

[6] T.Joachims, Making large-scale support vector machine learning practical, in: B.Scholkopf, C.J.C.Burges, A.J.Smola(Eds.),*Advances in Kernel Methods—Support Vector Learning*, MIT Press, Cambridge, MA,1999,pp.169–184.

[7] J.A.K. Suykens, J. Vandewalle," Least squares support vector machine classifiers", *Neural Process*. Lett. 9 (3) (1999) 293–300.

[8] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Publishing Co., Singapore, 2002.

[9] G. Fung, O.L. Mangasarian, "Proximal support vector machine classifiers", in: 7th *International Proceedings on Knowledge Discovery and Data Mining*, 2001, pp. 77–86.

[10] Jayadeva, R. Khemchandani, S. Chandra, "Twin support vector machines for pattern classification", IEEE Trans. *Pattern Anal.* Machine Intell. 29 (5) (2007) 905–910.

[11] O.L. Mangasarian, E.W. Wild, "Multisurface proximal support vector classification via generalized eigenvalues", IEEE Trans. *Pattern Anal.* Machine Intell. 28 (1) (2006) 69–74.

[12] Santanu Ghorai , Shaikh Jahangir Hossain , Anirban Mukherjee, Pranab K.Dutta , "Newton's method for nonparallel plane proximal classifier with unity norm hyperplanes," Elsevier. / *Signal Processing* 90 (2010) 93–104

[13] Santanu Ghorai , Anirban Mukherjee,PranabK.Dutta, "Nonparallel plane proximal classifier", Elsevier / *Signal Processing* 89 (2009) 510–522

[14] T. Evgeniou, M. Pontil, T. Poggio, "Regularization networks and support vector machines", *Advances Comput. Math.* 1 (13) (2000) 1–50.

[15] T. Evgeniou, M. Pontil, T. Poggio, "Regularization networks and support vector machines", in: A. Smola, P. Bartlett, B. Scho¨ lkopf, D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, 2000, pp. 171–203.

[16] G.Fung,O.L.Mangasarian, *SVM tool box home page* /http://www.cs.wisc.edu/dmi/svm/psvm.

[17] *LS-SVM toolbox,version-1.5advanced* /http://www.esat.kuleuven. ac.be/sista/lssvmlab/.

[18] *MATLAB,User'sGuide*,TheMathWorks,Inc.,1994–2001 /http:// www.mathworks.com.

[19]C.L.Blake,C.J.Merz,"UCI Repository for Machine Learning Databases", *Department of Information and Computer Sciences,* UniversityofCalifornia,Irvine,1998 /http://www.ics.uci.edu/ _mlearn/MLRepository.html.

[20]Black, Noel; Moore, Shirley; and Weisstein, Eric W. "Conjugate Gradient Method." http://mathworld.wolfram.com/ConjugateGradientMethod.html

[21]T.M. Mitchell, *Machine Learning*. Boston: McGraw-Hill, 1997.

[22]Y.-J.Lee,O.L.Mangasarian, "RSVM :reduced support vector machines", *Data Mining Institute*, Computer Science Department, University of Wisconsin, Madison, WI, Technical Report 00-07, July 2000,Available: /ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/ 00-07.ps.

[23]N.Cristianini,J.Shawe Taylor, *Kernel Methods for Pattern Analysis*, Cambridge UniversityPress, Cambridge,UK,2004

[24] S.Haykin, *Neural Networks—A Comprehensive Foundation*, seconded., Pearson Education, 2006, Chapter4,pp.235–240.